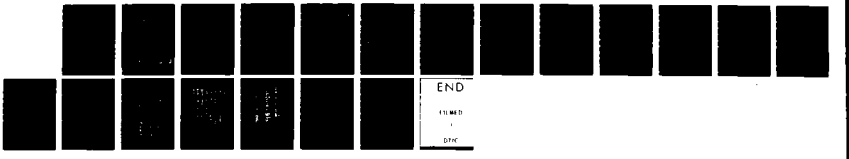
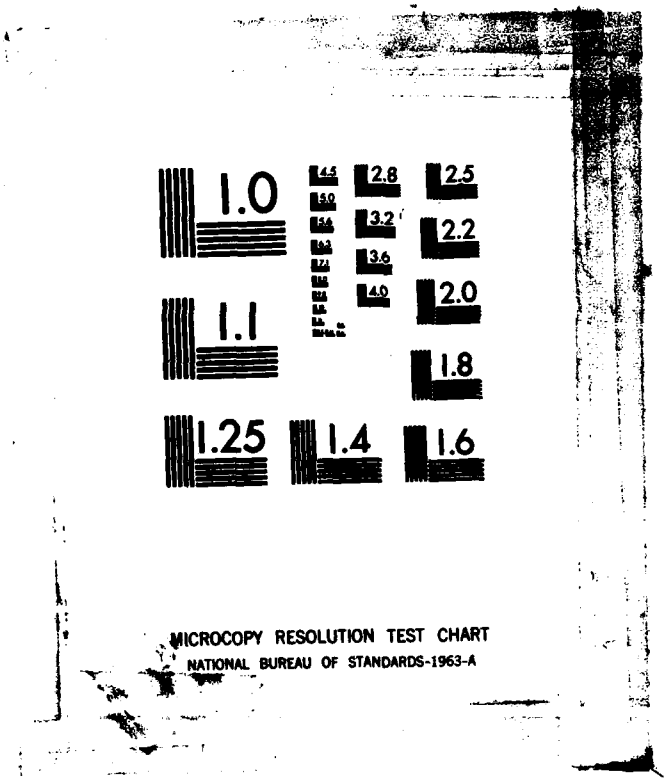
AD-A124 506 DIGITAL FLIGHT CONTROL SYSTEM VALIDATION(U) FEDERAL AVIATION ADMINISTRATION TECHNICAL CENTER ATLANTIC CITY NJ D ELDREDGE ET AL. JUN 82 DOT/FAA/CT-82/94 1/1

UNCLASSIFIED F/G 1/3 NL





DOT/FAA/CT-82/94

Digital Flight Control System Validation

Donald Eldredge

1982 American Control Conference
Arlington, Virginia

June 1982

Technical Paper

U.S. Department of Transportation
Federal Aviation Administration
Technical Center
Atlantic City Airport, N.J. 08405

DIGITAL FLIGHT CONTROL SYSTEM VALIDATION

Donald Eldredge
John E. Reed

Federal Aviation Administration

Ellis Hitt
Jeff Webb

Battelle-Columbus Laboratories

Dennis Mulcare
Lockheed-Georgia Company

ABSTRACT

The introduction of advanced technologies, new design concepts, and sophisticated high integrity integrated software-based digital flight control and avionics systems has confronted the FAA with the task of reviewing, revising, and updating its airworthiness assessment criteria in these areas. The FAA needs to establish and maintain systems engineering expertise as well as a capability to establish/evaluate validation/verification procedures for software-based digital systems. In order to accomplish this objective, the FAA must have expertise in fault-insertion techniques, emulation, automated reliability analyses, failure modes and effects analyses, fault-tree graphics, and other analytical tools. In addition, these skills are needed in order to interpret and evaluate data and information submitted, during the certification, process, in compliance with RTCA DO-178 and FAA Advisory Circular 25.1309-XX. This paper discusses techniques, methodologies, data, and information required for digital flight control and avionics systems validation.

INTRODUCTION.

The introduction of advanced technologies, new design concepts, and sophisticated high integrity integrated software-based digital flight control and avionics systems has confronted systems designers, integrators, and implementors with the need for developing an assurance methodology which can be used to properly validate the emerging systems. These new technologies, unlike previous analog and digital hardware, are being deployed in new system architectures which rely on digital buses for intercommunication and exchange of data and information. They are being applied to flight critical functions in single string and redundant designs, using monitoring schemes and redundancy management techniques, to insure the essential safety of these systems whose authority may not be limited, in terms of control. Moreover, the quality of the software in these systems may directly impact safety of flight.

The high levels of performance, functionality, criticality, and complexity associated with the new integrated systems require that each hardware and software component, module, or subsystem must be more thoroughly verified and validated before being integrated into the final product and the Operational Flight Program (OFP) being exercised in the aircraft. The verification and validation methodologies must employ testing, emulation, simulation, analytical methods, and configuration management at much higher and more critical levels than ever before. The safety assessment of these systems requires substantiation through analysis and where necessary by appropriate test (test meaning actual execution and data observations). The hardware, software, and integration of hardware and software must be evaluated to prove that a combination of failures (due to hardware failure or software problems), that could prevent the continued safe flight and landing of the aircraft, is improbable or highly improbable. The probabilities of failure associated with these analyses are specified in terms of probabilities in the range of 10^{-5} to 10^{-9} or less. To achieve these goals, it is required that the critical components within the system (CPU, RAM/ROM, A/D's, etc.), the subsystem modules (card level), and the subsystems be screened and then quality controlled. The software must be developed and maintained in such a manner that the possibility of an unintended function or nonconsequencing code is minimized (see table 1). The regulatory agencies in recognition of these criteria have specified the probability classifications in several documents (references 1-4). These documents constitute valuable guidelines for assurance assessment of essential/critical digital systems and provide the framework for the establishment of formal and informal methodologies for the verification and validation of digital flight control and avionics systems.

TABLE 1. QUANTITATIVE SAFETY REQUIREMENTS

Frequency of Occurrence	Probability Range (Events per hour)	
	FAA (AC 25.1309)	JAA (JAR 25.1309)
Probable	$\leq 10^{-5}$	10^{0} to 10^{-5}
Frequent		10^{0} to 10^{-3}
Reasonably Probable		10^{-3} to 10^{-5}
Improbable	10^{-5} to 10^{-9}	10^{-5} to 10^{-9}
Remote		10^{-5} to 10^{-7}
Extremely Remote		10^{-7} to 10^{-9}
Extremely Improbable	less than 10^{-9}	less than 10^{-9}

Problem.

The development of digital flight control and avionics systems for commercial transport aircraft has progressed very rapidly and in a revolutionary manner in the last 5 years. The revolutionary manner has been adopted due to the rapid proliferation and availability of microprocessor and large-scale integrated circuit (LSI) technology. This technology, coupled with the advent of structured higher order languages and formal programming logic and design, has resulted in the accelerated development of high technology systems. These systems not only duplicate the functions of the previous analog and hybrid systems, but add more functions and capabilities such that the new systems outperform the previous systems and, in fact, exercise more control and authority over the transport aircraft than ever before.

In addition to the revolutionary application of technology due to availability, the new technology has also been adopted due to the economic pressures and considerations that prevail today and can be expected to continue through the year 2000. The relative economic costs, (due to interest rates, maintenance, fuel, labor skills, component reliability and availability, taxes, and the need to make a profit) have increased the demand to apply the new technology in an effort to lower operating costs, increase reliability, functional readiness and availability, decrease the weight of the avionics systems and the aircraft itself, and to provide more functions which can be expanded and changed without the need to purchase additional hardware components. The above goals, in order to be reached, rely heavily on the discipline of Software Engineering and the ability of the innovators, developers, integrators, and manufacturers to come to terms with software design, implementation, testing, and maintenance. The previous technology (analog, hybrid, and digital) relied heavily on past experience gained through many years of design, implementation and maintenance; therefore, the risks associated with developing a new system or subsystem were well known and the failure rates could be predicted. With the new integrated systems, however, the technology is new and the risks are difficult to estimate, and in fact the assurance assessment methodology has not yet matured to the point where the verification and validation of the new systems has been shown to be totally effective.

Thus, the issue of public safety in the new generation of commercial transport aircraft, which employ digital flight control and avionics systems, is of great concern to the manufacturers, operators, certificators (regulators), and the general public--who are the users. The concern over the issue of public safety, however, need not result in the implementation of regulatory actions resulting in rules/regulations which excessively restrict the system innovators, developers, integrators, or manufacturers ability to be flexible in design procedures, production processes, maintenance practices, and in the use of innovative technology. It should, however, result in a closer relationship between regulatory and industry personnel which can result in the establishment of guidelines, techniques, and methodologies which may be used in order to assure that the resultant systems are performing their intended function and indeed safe. (Such was the case in the formation of the Radio Technical Commission for Aeronautics (RTCA's) Special Committee SC-145.)

The RTCA SC-145 Committee generated Document Number RTCA DO-178 (reference 5) which is intended to provide guidance in the development and certification of software in airborne systems. This document, however, is the first of many such guideline documents which must be produced in order that the proper techniques, tools, and methodologies may be understood by both the regulators and industry; and that the development, implementation, and validation of these new systems takes place in such a manner that system safety is assured at a very high level.

The certification procedures for new aircraft and aircraft systems are governed within the FAA by Orders 8110.4, "Type Certification," and 8110.8, "Engineering Flight Test Guide," various advisory circulars, the RTCA Documents, Minimum Operational Performance Standards (MOPS) and Society of Automotive Engineers (SAE) Aerospace Standards/Aerospace Recommended Practices, Technical Standard Orders, and Special Conditions. These documents are minimum requirements and guidelines which are applied to Type Certification and Supplemental Type Certification activities involved in a process which starts with technological innovation and results in the in-service implementation of a safe and reliable product developed under economic constraints, technology initiatives, and concern for public safety. In order to effectively accomplish all of this, government and industry personnel must be aware of and should make use of the current tools and methodologies available for verification and validation of digital systems within the context of the certification process. It is particularly important that the new "software" element be verified through the use of the proper assurance methodologies, and that the manufacturer, the quality assurance specialists, and the certification engineers understand the results of the application of these tools and methodologies and use them in the development and certification processes.

Agents/Roles.

In order to accomplish the design, development, and integration stages of the hardware and software subsystem, the manufacturer translates the concepts

and requirements into specifications which are ideally furnished simultaneously to the developer of the hardware and software as well as to the testing organization(s). In addition, a similar set of specifications and data packages are submitted to the certification specialists in response to the type certification or supplemental type certification requirements. The specifications and data packages submitted for certification should include test plans, test procedures, and test reports which have been developed by the testing organization. The data package delivered by the testing organization should include not only the data derived from the tests, but also the software drivers, test control software, data collection software and data analysis software. The agents, their roles, and the interaction of these activities are presented in figure 1.

Current Tools and Methodologies.

Table 2 outlines the currently available tools and methodologies which may be used for verification and validation of digital flight control systems. The table presents the tools which are available for Reliability Assessment and for Performance and Safety Assessment. Furthermore, the table also presents the expected results (based upon the application of these tools) and the regulatory (or guidance) documents whose compliance may be necessary for the certification process. These tools and methodologies are usually applied within the framework of a life cycle development process. The software embedded in the digital flight control system is evaluated throughout the life cycle stages in such a manner that the verification and validation activities address compliance with the software specifications in terms of correctness of implementation; and that the required assurance levels (in terms of probability of unintended function occurring) are attained. The methodologies identified and utilized in the verification and validation process need to be comprehensive with regard to scope of software error detection and coverage for the various stages of the software life cycle as identified in figure 2 and table 3. Furthermore, it is important that the verification and validation process should occur in each phase of the cycle rather than in a single isolated stage following construction, because problem statement or design errors discovered that late may exact an exorbitant price. Not only must the original error be corrected, the structure built upon it must be changed as well (references 6 and 7).

TABLE 2. CURRENT TOOLS AND METHODOLOGIES FOR VERIFICATION AND VALIDATION OF DIGITAL FLIGHT CONTROL SYSTEMS

TOOLS	METHODS OF UTILIZATION	EXPECTED RESULTS	REGULATORY (GUIDANCE) DOCUMENTS
FAULT TREE ANALYSIS	• MANUAL TECHNIQUES • COMPUTER	LOGIC TREES GENERATED	(DEPENDENT COMMERCIAL PROGRAMS)
FMEA	SYSTEM/SUBSYSTEM/COMPONENT FAILURES	IDENTIFICATION OF FAILURE & EFFECTS OF FAILURES	• FAR's 25, 27, 29 • MIL-HDBK-217C • AC 25-1309-1A • MIL-P-87939
ADVANCED RELIABILITY	• CARA • CARE III • ARIES, ETC. • QUALITATIVE ASSESSMENT	} MODELS	PROBABILITY OF SINGLE FAILURE (CRITICAL)	• FAR's 25, 27, 29 • AC 25-1309-1A • NASA TM-X78
FAULT TREE	• FAULT TREE GRAPHICS • K-FAULT, ETC. • TOP LEVEL FAILURES (TOP EVENT)	• LOGICAL MODE OF FAILURES WITH RELIABILITY PREDICTIONS	• FAR's 25, 27, 29 • MIL-HDBK-338B • ARP-4754 A/C • AC 25.1309-1A • NASA-CR-172
ZONAL (CONCEPT) ANALYSIS	• MANUAL SOFTWARE ANALYSIS • MANUAL CONCEPT ANALYSIS	COST-EFFECTIVE/ INDEPENDENT ASSESSMENT OF ALL IDENTIFIED FAULTS	(DEPENDENT COMMERCIAL PROGRAMS)
SIMULATION	• HOST COMPUTER • DEVELOPMENT CENTER	• EVALUATE THEORIES OR PROPOSED VALUES SIMPLE DESCRIPTION OF LOGICAL STRUCTURES OF A SYSTEM	(COMMERCIAL SYSTEMS/ PROGRAMS)
SIMULATION	• HOST COMPUTER • TARGET SYSTEM • PROGRAMS • DYNAMIC • FAULT INSERTION	• FLYING QUALITIES • HANDLING • FAILURE EFFECTS/ RECOVERY PROCEDURES	• FAR's 25, 27, 29 • AC 25.1309-1A • AC 25.1309-1A
ENVIRONMENT	• VIBRATION • ATMOSPHERE • SHOCK, ETC.	LIFE CYCLE/ FAILURE RATES	• MIL-STD-810D • FAR's
AIRCRAFT	• TEST BED AIRCRAFT • PRODUCTION AIRCRAFT	• FLYING QUALITIES • HANDLING • TYPE CERTIFICATION	• FAR's (INSTALLATION • FAA ORDER 8110.4 • FAA ORDER 8110.4 FAR's 25, 27, 29 MIL-F-8785C

TABLE 3. LIFE-CYCLE VERIFICATION ACTIVITIES

Life-Cycle	Verification Activities
Requirements	Determine Correctness and Consistency Generate Functional Test Data
Design	Determine Correctness Determine Consistency both Internal and With Previous Stages Generate Structural and Functional Test Data Refine Existing Test Data Generated Earlier
Construction	Determine Correctness Determine Consistency both Internal and With Previous Stages Generate Structural and Functional Test Data Apply Test Data Refine Existing Test Data Generated Earlier
Operation & Maintenance	Same Refine Existing Test Data Generated Earlier

METHODOLOGY DEVELOPMENT.

In order to insure that the software in the digital flight control system complies with the definition and design criteria in the software specification (s), and/or that it was correctly implemented, it is necessary to establish a methodology (or set of methodologies) for assurance assessment that is coupled to software function criticality, implementation complexity, and thoroughness of testing. The development of this methodology must be based on the latest software engineering technology, "good engineering judgment," mature technical skills, manual and automated static/dynamic software testing tools, failure modes and effects analysis, and reliability analysis, and as such, will result in the implementation of a hardware/software validation methodology which will establish the correctness of the hardware/software implementation in a systematic and conclusive manner. The methodology, when finalized, should cover all seven of the development phases and the 25 activities identified by Wolverton (reference 8) and summarized in figure 3. In addition, the methodology must be keyed to the characteristics of the system being verified and validated such that the types of possible software errors which might be present in the code are taken into consideration. "The nature, source, and likely time of occurrence of these errors are then examined to determine the software error detection coverage required according to the stage of the development process" (references 6 and 7).

The development of an assurance methodology keyed to the characteristics of the system under development requires: (a) That the system characteristics are well understood; and (b) That the associated verification and validation concerns be delineated such that the impact on the software is well understood and testable. Table 4 presents such a relationship and displays the interaction of system considerations, impact on software and verification and validation concerns (reference 6).

TABLE 4. DIGITAL FLIGHT CONTROL SYSTEM CONSIDERATION AND VERIFICATION/VALIDATION CONCERNS

[illegible] CONSIDERATION	IMPACT ON SOFTWARE	V&V CONCERNS
System Functions	. Operating System . Multiple Tasks . Microprogrammed Instructions	. Top Level control & Start-up . Mode Logic & Initialization . Redundancy Applications Use
System Interfaces	. Sensor Inputs . Cross-channel Data Transfer . Servo Command Outputs . Bus Control	. Software Fail, Aliasing, Runstuff . Software Fail, Skewing, Signal Select . Software Fail, Aliasing, Commissing . Software Fail, Utility Functions, Processin
Timing	. Real-time Execution . Channel Synchronization . Throughput . Background Functions	. Subsystem Throughput . Channel Skewing . Every Cycle Completion . Adequate Servicing
Stability	. Transport Lag . Input Sample Rate . Control Law Execution	. Minimal Discontinuous Delay . Data Dispersion . Acceptable Phase Lag
Redundancy	. Execution Rates . Difference Equations	. Data Limitations . Coefficient Sensitivity
Functional Reliability	. Channel Replication . Analytical Redundancy . Resident Software	. Separate Software Drivers . Background Algorithms . Dissimilar Channel Combination
Integrity	. Fault Detection . Self Monitoring . Self-Test . Equalization	. Comparison/Voting Algorithms . Coverage, Effectiveness . Test Intervals, Coverage, Effectiveness . Valid Cross-Channel Monitoring
Human Interface	. Control/Displays . Pilot Commands . Status Annunciation	. Prioritize, Refresh Rates, Formats . Clarity, Timeliness, Effectiveness . Clarity, Augmented Decision
Operational Acceptability	. Handling Qualities . Atmospheric Conditions . Workload	. Varied Flight Dynamics Dependencies . Turbulence Response, Crosswind, Signal Noise . Composite Tasks, Anomaly Procedures
Physical Effects	. Sensor Noise . Electromagnetic Interference . Electrical Power	. Aerodynamic, Electrical, & Servo Vib Noise . Lightning, Communi or Induced Interference . Power Transients, Power-on Start-up

The reviews and walkthroughs are used quite frequently and, in general, verify that each level of life cycle (phase) satisfies the requirements of the previous level. The analytical tools, such as proof of correctness and symbolic execution, are not "mature" or well developed for this generation of software systems, but they may have an increasing role in the future.

The automatic verification methods are classified as either static or dynamic analysis, depending on whether they involve execution of the software code with real-time data. Static analysis involves execution on a host computer of an analysis tool which accepts as input the software to be tested, and as such, examines some aspects of the specifications, design, or code in an analytical environment. The most commonly used static tool is the compiler/ assembler which tests for language construct useage. Other tools include data flow and control flow analysers that are language specific or must have specially embedded comments. In addition to the compiler oriented tools, Sneak Software Analysis can be used to identify possible faults without the failure having to occur (reference 9).

Tools and Techniques.

The tools and techniques used in a software assurance methodology can be classified as manual or automatic, and can take place at the module, integration, and system levels.

Manual verification methodologies are applicable to all life cycle phases, although they are generally applied during the design and implementation (construction) phases. These methodologies include:

Desk Checking and Review.
Structured Walkthroughs and Inspections.
Formal Methods of Proof and Correctness.
Symbolic Execution.

Dynamic tools perform some functions which aid in the testing or assessment of executing software. Dynamic analysis basically observes the target software as it is executed to determine coverage. Coverage includes: statement execution, all possible branch paths, boundary analysis, and test coverage. The test case design process is vital in order to minimize the number of required tests to exercise the target software and to determine the test coverages.

The currently available static and dynamic tools are presented in table 5. It will be noted that the static tools are partitioned into the categories of specific and general. Each of the specific static tools examines a particular property of software on a direct basis, while the general static tools address more general properties in a more adaptive manner.

TABLE 5. CURRENTLY AVAILABLE STATIC AND DYNAMIC TOOLS

Specific Static Tools	General Static Tools	Dynamic Tools
Cyclomatic complexity checker	Accuracy analyzer	Simulators
Code comparator	Assembly code verifier	. Computer . Hybrid . Test bed (live data) . Monte Carlo
Conditional checker	Assertion checker	
Cross-reference checker	Documentation and documentation systems	Test data generator
Data base analyzer		
Flow checker	Formal languages with system analyzers	Test driver
		Test execution monitor
Interface checker	. Interconnection . Specifications . Program design . Program code	Test record generator
Program flow analyzer		Timing analyzer
Set use checker		
	Stack-path analyzer	
Standards checker		
	Symbolic evaluator	
Table consistency		
	Thruput power	
Instrumentation code developer		
	Verification condition generator	

The automatic verification methodologies are also applicable to all life cycle phases although they are generally used for testing modules and submodules at the unit level.

Integration Level Testing.

After successful unit level testing, the units (modules) are connected to determine whether they function together in tandem. Integration testing is defined as the verification of the interfaces among system parts, such as: components, modules, and subsystem. Integration testing treats the software at the component level rather than at the detailed level of code that was the subject of unit testing. Thus, the main testing emphasis is on the interaction between software components and their interfaces. There are a large number of possible approaches that can be used to sequence the testing of modules and the merging of modules into larger entities. Most of the approaches can be described as variations of one of seven basic approaches. The seven approaches are:

Bottom up testing
Top down testing
Modified top down testing
Big-Bang testing
Sandwich testing
Modified sandwich testing
Thread testing

A further distinction is sometimes made between incremental and phased implementation methods, but the above seven techniques encompass these methods.

Table 6 presents the methods and key characteristics of module integration testing strategies.

TABLE 6. MODULE INTEGRATION TESTING STRATEGIES

Method	Key Characteristics
Bottom-Up Testing	. Program is merged and tested from the bottom to the top . Only terminal modules tested in isolation . Requires module drivers
Top-Down Testing	. Begins at the top of the program structure and proceeds to test components at progressively lower levels in the hierarchy . Requires dummy modules, or stubs . Combines module testing, integration testing, and a small amount of essential function testing
Modified Top-Down Testing	. Same as top-down testing, but requires each module to be unit tested in isolation before it is integrated into the program
Big-Bang Testing	. Every module is unit tested in isolation . All modules integrated at once . Requires both stubs and module drivers . Debugging is difficult
Sandwich Testing	. Compromise between bottom-up and top-down testing . Testing begins from both top and bottom simultaneously, and proceeds until meeting in the middle . Difficult to apply to small programs . Requires both stubs and module drivers
Modified Sandwich Testing	. Modules in upper levels of the program are unit tested in isolation . Proceeds like Sandwich Testing
Thread Testing	. Useful in demonstrating the operation of certain important functions early in the testing process . Allows testing and analysis in digestible quantities . A thread is defined as a string of program units which accomplish a distinct processing function

System Level Testing.

The end objective of all testing and validation activities is to ensure that the delivered software product satisfies all specified functional and performance requirements and the identified design objectives (reference 10). If test cases are selected strictly with these high-level objectives in mind, then the set of test cases is not necessarily representative of the anticipated operational usage, in which case, reliability of the software is not necessarily demonstrated nor guaranteed. It is a frequent occurrence to find software errors during operational use that were not discovered during testing because no test case ever exercised certain sections of code (reference 11).

System testing is probably the most misunderstood form of testing. Function testing is the testing of all functions of the completely integrated system. System testing is the process of trying to find discrepancies between the system performance and its original objectives. System testing is a validation process when it is done in the end-user's actual environment. However, when the actual environment is not available, system testing is a validation process which is performed in a simulated or test environment (reference 10).

Systems testing often requires more creativity than the testing methods described previously. Designing good system test cases may require even more ingenuity than was required to design the system. Many of the types of tests which may be necessary during system testing are summarized in table 7.

TABLE 7. TYPES OF SYSTEM TESTS

Name	Description
Load/Stress Testing	Subject the system to extreme pressures.
Volume Testing	Expose the system to massive amounts of data over a long period of time.
Configuration Testing	Test the software configuration.
Compatibility Testing	If the system is used with an existing system, compatibility testing is required.
Storage Testing	Tests main and secondary storage objectives.
Performance Testing	Tests performance or efficiency objectives.
Installability Testing	Tests the installation process of a system.
Reliability/Availability Testing	Shows that the system does or does not meet its original reliability.
Recovery Testing	Tests the recoverability of a system.
Serviceability Testing	Tests system serviceability and maintainability.
Human Factors Testing	Brings end user into the testing loop.

The choice of tools and techniques used in the assurance methodology at the module, integration and system levels will be, in part, dictated by the complexity and criticality of the software being tested, as well as the requirements imposed by the certification process. The end goal, of course, is a digital flight control system is the identification and elimination of all errors at each stage, and ultimately, confirmation of their absence.

The application of testing and analysis at the module, integration, and system level during the design and construction (coding) phases will not necessarily produce a "perfect" software system; however, it can be inferred that, if a methodology is adopted and carried out, "high quality" software will be the result. Furthermore, if the testing organization is successful in implementing an integrated package of tools and techniques, the verification and validation will be accomplished at high levels of assurance and at the least cost.

Error Management and Redundancy.

In addition to the development and testing methodologies established during the design, construction (coding), and testing phases, the assurance assessment of a digital flight control system may require that the system be designed such that error management, monitoring for failure conditions, and redundancy be included in the various implementation stages. "The need for, and extent of, failure monitoring is determined from the nature and architecture of the system, and its requirements and criticality. When needed, such monitoring may be accomplished by using either, or a combination of, the following approaches:

- "Minimal Redundancy" (Independent and identical subsystems), with comparison and/or voting.

"Various types of self-testing/built-in-test (BIT) within each individual system or subsystem."

"Either method, if thorough and comprehensive, can provide for "FAIL-PASSIVE" or "FAIL-OPERATIVE" critical system operation meeting the appropriate probabilistic requirements of regulations such as FAR 25.1309."

"A possible methodology for the management of both ERRORS and FAILURES is presented in figure 4."

"Self-testing (BIT) in individual systems or subsystems can generally be sufficiently comprehensive only for computerized implementations. When used, the following techniques (or their equivalents) are commonly employed" (reference 12):

CPU Diagnostic
Program Memory Checksum
Ram Memory Check at Power-Up
Iteration Monitor

Cyclic/Longitudinal Redundancy Checks
Power Supply Monitor
Error Routine
Reasonable Test
Wrap-Around Test
Input Vector
Cross-Channel Monitors
Feedback of Control Surface/Servo Position
Performance, Envelope, or Limit Monitoring
Others as Appropriate/Necessary

At the present time, the effectiveness of each of these monitors (built-in test/fault isolation test) has not been established in terms of percent of coverage, nor has the relative cost of implementation of these techniques, in a digital flight control system, been established. However, it may be inferred that, if a system is developed, tested, and implemented at the component, module, subsystem stages without "some level" of monitoring coverage, in the in-service system, the probability of detecting and recovering from a single "potentially catastrophic" failure will be decreased (reference 12).

Failure Effects and Reliability.

The emergence of digital hardware as the primary medium for flight control has paralleled a trend of increasing the number and criticality of functions performed by the flight control system. These factors have increased the need for personnel with design, managerial, and regulatory responsibilities to be familiar with reliability and failure effects prediction methods and their application to digital systems.

The need for selection of state-of-the-art methods is driven by the urgency of near-term digital flight control system (DFCS) validation requirements. The number of such systems approaching operational use makes it appropriate to assign higher priority to their analysis. The method selection for the verification and validation process is based on a set of criteria which reflect method needs and usability. In the establishment of a methodology, it is desired to include (and analyze) the following:

Hardware reliability
Hardware failure effects
Software reliability
Software failure effects
Man-machine interface
Design faults

A number of methods address the first two factors (i.e., hardware reliability and hardware failure effects). Although a number of software reliability models exist, they are not as well developed (reference 13). In general, these models are based on the hypothesis of: (a) The relation between the number of errors in the software originally; (b) the number remaining; and (c) the rate at which errors are detected and corrected. These models can be useful in large software development projects; however, their credibility for predicting very low failure rates has not been established (i.e., in the order of 10^{-5} to 10^{-9}) at the present time.

Failure effects methods for software are even less developed than software reliability methods, and design faults in the development and manufacture of the LSI/VLSI Chips, which may be important in very large-scale integrated circuits, may not have been included in existing reliability models.

The development of automated tools (for assessing software errors, the man-machine interface, and design faults), is progressing, however, the existing tools may limit the scope of the method selection process to existing reliability and failure effects methods which are limited in their capability to fully examine all of the interactions that may be important.

Reliability.

The term "reliability analysis" is used in a broad sense to refer to evaluation of the capability of a system to perform its intended function for a specific interval under stated operational and environmental conditions (reference 1). The intended functions for DFCS are defined in terms of sensor, servo, and computational requirements for safe flight. Reliability analysis considers the rate at which failures occur and the impact of a particular failure on system performance. "Reliability" relates to the probability of a failure occurring at specific points in time, while "failure effects" are the results of specific failures.

The probability of airplane failure due to failure of the DFCS is required to be extremely improbable. "Extremely improbable" is usually taken to mean the probability of catastrophic DFCS failure, and is on the order of less than 10^{-9} per hour; while improbable events have a frequency of occurrence in the range of less than 10^{-5} per hour (reference 1). Computations to this level of accuracy require special considerations.

System success and failure paths consist of various combinations of good and bad components, modules, and subsystems. Combinations of individual failures must be considered when analyzing a fault tolerant system design. As a result, system failure probability consists of many products and sums involving numbers close to one and numbers close to zero.

Numerical accuracy in such computations can be lost very quickly due to round-off errors. In addition, transcendental functions (e.g., exponentiation,

logarithm) are computed using truncated series approximations. The accuracy of these approximations may very from machine to machine.

Reliability analysis of DFCS requires rigorous numerical accuracy in order for the results to be useable for the specification of the probability that the system will perform its intended function. Assuming sufficiently accurate data is available, the analysis technique should be capable of maintaining accuracy in the computations.

The methods used for predicting reliability and failure effects must:

Provide the capability of assessing the desired probability of safe flight to the accuracy required (10^{-5} to 10^{-9} per flight hour).

Properly treat all failure modes of complex DFCS architectures designed to interface with the aircraft electrical systems.

Include all failure effects associated with digital technology including software.

Include treatment of all of the following faults:

Physical faults
Transient faults
Permanent faults
Local (single) faults
Distributed faults
Determinant faults
Indeterminant faults
Development faults
Interaction faults

Implicit in the requirement for models that accurately predict the reliability and failure effects for these types of faults (in the system architectures which may be developed), is the need for data which will be input to the model. Generating and obtaining good component and module failure data is a well-recognized problem in reliability analysis of electrical systems. In many cases, approximations and "best case" and "worst case" analyses have been deemed sufficient. However, the critical role of DFCS in airplane safety means the system failure probability must be quite small (i.e., 10^{-5} to 10^{-9} per hour). Several mathematical models have been designed for computing the failure probability of complex systems. However, accurate output requires accurate input. No matter how accurate the model may be, inaccurate input data will result in an inaccurate estimate of system reliability.

It is, therefore, imperative that accurate data be collected on digital equipment used in flight control systems. The data should describe:

Permanent faults
Intermittent faults
Transient faults
Latent faults
Error detection methods
Error recovery schemes.

Without accurate and meaningful data, it will not be possible to compute system failure probabilities to the desired order of magnitude. Due to the high probability of safe flight which the DFCS must provide, the data describing these systems must be accurate and include the distribution of the failures as well as the "expected value" since the tails of the distribution are important (reference 14). Mareliez and Bjurman (reference 15) have outlined an approach for application of "reliability analysis" to an autoland system containing control wheel steering (CWS) and Category II and III capability, and defined the reliability requirements. Their approach is as follows:

"The following assumptions, addressing the reliability assessment of this type of function, are deemed pertinent:

"(1) The aircraft will not be dispatched on a revenue flight unless the CWS function is completely fault-free.

"(2) The exposure time on which to base prediction of average system failure probability, or average hazard risk, will be the average flight duration of jet transports today, or approximately 1-hour.

"(3) The exposure time on which to base the prediction of specific system failure probability for a long flight will be 10 hours.

"(4) The system must provide failure status indication in time to allow for a minimum 30 minute diversion.

"(5) Autoland function failure (complete loss of function) is assumed to be hazardous under two conditions: (1) It occurs during the 45 seconds following alert height passage in Category III conditions; or (2) it occurs during the same 45 second phase of Category II or better visibility, and the pilot fails to recover control. The combined probabilities of autoland failure and Category III weather, or autoland failure and pilot failure to recover, respectively, must be shown to be equal to or less than 1×10^{-6}.

"(6) Allowing for a 1 percent probability of Category III weather conditions, which appears to be a conservative assumption for a worldwide average based on existing weather statistics, the result is a required autoland function failure probability equal to or less

than 1×10^{-7} per landing. Probability of pilot failure to take over in Category II or better visibility of one in 1,000 has been accepted and used for certification purposes; this results in a required autoland failure probability of 1×10^{-6} or less, in other words a less demanding design condition than Category III.

"(7) Requiring an autoland function failure probability of 1×10^{-7} or less results in accidents due to autoland function failure being extremely improbable; less than one accident in present day jet transport accident data show approximately no fatal landing accidents (from all causes) per one million landings" (reference 15).

Various techniques exist for evaluating the effectiveness of aircraft computing systems. These techniques have been used for assessing primarily the reliability and safety of flight control systems and digital avionics. The techniques are generally mathematical models which may be manually applied or may be implemented in computer programs. These models are normally used rather than testing techniques to determine the reliability due to the cost of performing reliability testing.

With the development of fault tolerant computing systems, testing becomes even more impractical due to the many fault tolerant architectural concepts which are possible, and the fact that testing requires that the system design be committed to hardware and software. Techniques are required which can be used to design fault tolerant computing systems as well as evaluate the design of candidate fault tolerant computer systems prior to the actual development of the hardware and software which implement the candidate design.

The total system must be analyzed and not just a portion such as the hardware components or the software. The nature of the systems to be analyzed are categorized by the complexity of relationships among system elements under the control of a software executive program. This complexity can lead to an intractable analysis problem for completely general system. Moorelian and Bjurman (reference 15) in recognition of the above requirement, developed an analytical reliability model called Computer-Aided Redundant System Reliability Analysis (CARSRA) to handle the analysis of fault tolerant modular redundant flight control systems.

The Fault Tree Method.

Fault trees have been widely used in many types of reliability analysis, since development of the technique in the early 1960's. The technique is conceptually quite simple, though application to realistic problems may be laborious.

There are two aspects of the methodology which will be discussed separately:

(1) Construction of the fault itself; and (2) computation of the probability of the events considered. In some applications, only the first aspect is used. In the present study, both are required.

Construction of the Fault Tree.

The starting point for each fault tree is the selection of some particular event (usually an undesirable event) for study. In most problems there is more than one type of failure to be considered. In such cases, a separate fault tree must be developed for each type. Examples would be:

(1) Loss of aircraft through control failure.
(2) Loss of all aircraft position information.
(3) Loss of Category II landing capability.
(4) Loss of area navigation (RNAV) capability.

This fundamental event to be studied is sometimes designated as the "Top Event," since it occurs at the top of the fault tree as usually drawn.

Once the top event has been selected, the next step is to enumerate all the ways in which the top event can happen. This enumeration is done through use of a specific type of graph structure known as a tree, hence the name fault tree.

The construction of a complete fault tree proceeds from the top down. The top event is defined, and those events leading to the top event are defined. Then those events leading to the events just below the top event are defined.

This continues until a level of fundamental events is reached. The nature of this fundamental level can be selected for the purposes of a particular problem. It could be failures of fundamental components such as resistors or solder joints. It could be failure of major subsystems, such as an inertial navigation unit or a particular computer function.

In concept, this is all there is to the construction of fault trees. In considering specific problems, however, considerable ingenuity may be required to fit the problem into this framework. For example, in the fault tree, there is no explicit recognition of time. This can be overcome, at least in many cases, by time-related definition of events. For example, a top event could be defined as loss of control during a specific period of time, such as final approach and landing. If there is more than one time period of interest, it may be necessary to construct a different fault tree for each time period, and for each top event in each time period. Conceptually, this is a

simple approach, but the labor involved in constructing many fault trees should be considerable.

Also, it is necessary for the analyst to have a very good understanding of the system being analyzed. It is important that all ways of reaching the top event be portrayed in the tree. There is no general way to assure this, but the more the analyst knows about the working of the system, the less likely the analyst is to overlook failure-producing events or combinations of events.

Determination of Failure Probabilities.

Once a fault tree has been developed, it may be desirable to determine the probability of the top event. In order to do this, it is necessary to know the probabilities of the fundamental events. If the fundamental events are independent, the determination of probabilities is relatively straightforward.

Probability computations start at the bottom of the tree with the fundamental events and proceed upward, using formulas at each stage until the top event is reached (reference 16).

In the case of a digital flight control system, the analysts computed the following mission outcomes:

(1) Probability of successful on-time landing at the original destination.

(2) Probability of successful, but late, (based on flight management loss prior to landing phase) landing at the original destination.

(3) Probability of diverting and safely landing at the alternate destination.

(4) Probability of aborting (due to loss of all spares with only a remaining and safely landing at the origin.

(5) Probability of loss of aircraft during the mission; and probability of loss of aircraft during each phase (reference 14).

ANALYTIC SUMMARY.

The analytical solution of the dual-dual system problem uses three model levels — mission, function, and component — in addition to the accomplishment set. A concept of "independence with respect to mission outcome" is used to accommodate the large number of trajectories in the base (i.e., component level) model. Probability computations are performed using the matrix multiplication procedures to analyze the dual-dual system.

The accomplishment set is $A = a_0, a_1, a_2$ where the a_i represent specific mission outcomes (i.e., accomplishment levels) of interest. In particular,

a_0 represents safe flight and successful landing at the primary destination;

a_1 represents safe flight and successful landing at the alternate destination;

a_2 represents loss of the aircraft (unsafe flight or unsuccessful landing) (reference 14).

The fault tree for "loss of control" in a dual-dual configuration is shown in figure 5. This fault tree is one of many top-level fault trees that would be developed in order to estimate probabilities associated with: (a) The successful completion of the mission; (b) partial completion of mission; and/or (c) loss of control resulting in "loss of aircraft" (reference 14).

The probabilities (of failure) associated with each level of the logic tree are used in the computation of system reliability and functional readiness which are calculated in the content of a "stage Markov model" such as the CARSRA model developed by Mastaliez and Bjurman (reference 15).

The "stage Markov model" analyzes stages which include:

Permanent faults
Transient faults
Coverage
Detected Stage Failures
Undetected Stage Failure

An example of a stage Markov model is presented in figure 6 and reference 17. The "stage Markov model" when completed outputs the data in table 8.

TABLE 8. STAGE MARKOV MODEL

I Functional Readiness Computations

$FR_i(t_1)$ = [PROBABILITY FUNCTIONAL READINESS CONFIGURATION i EXISTS AT TIME t_1]

$FFR(t_1)$ = FUNCTIONAL READINESS = $\sum_{i=1}^{n} FR_i(t_1)$

$FFR(t_1)$ IS OUTPUT BY CARSRA FOR EACH SPECIFIED TIME t_1

II System Failure

$FF_i(t_2)$ = PROBABILITY [SYSTEM FAILS BY TIME t_1+t_2 GIVEN FUNCTIONAL READINESS CONFIGURATION i AT TIME t_1]

$FFF(t_2)$ = PROBABILITY [SYSTEM FAILS BY TIME t_1+t_2 GIVEN FUNCTIONALLY READY AT TIME t_1]

$$= \left(\sum_{i=1}^{n} FF_i(t_2)\ FR_i(t_1)\right) \Big/ FFR(t_1)$$

$FFF(t_2)$ IS OUTPUT BY CARSRA FOR EACH SPECIFIED TIME t_2 AND EACH FUNCTIONAL READINESS TIME t_1

III Mission Failure

MF = PROBABILITY [SYSTEMS DOES NOT MEET ANY FUNCTIONAL READINESS CONFIGURATION OR FAILS AFTER THE FUNCTIONAL READINESS TIME]

$MF = [1-FFR(t_1)] + FFR(t_1)\ FFF(t_2)$

These analytical results, coupled with the detailed completion of the fault trees, should result in an accurate estimation of the component, module, and system availability and potential failure rates based on a system's approach to evaluating the effectiveness of the computing systems/subsystems in a digital flight control system (reference 17).

Currently available analytic models, such as CARSRA and fault trees can be used as tools to assist in the design, engineering development, and certification of digital flight control systems and the specification of the failure mode and overall system reliability. Assessment of system reliability requires assessment of hardware operational faults, design faults, software errors, and man-machine interface faults. Analytic models and methods can provide one aspect of the total equation. The application of these models assure that the software at the module, integration, and system levels have been adequately tested and are free from error.

In the design phase, models and methods can be used to evaluate the impacts of candidate system architectures and fault-tolerance techniques. Sensitivity analysis can be performed to assess the impacts of variations in levels of redundancy and in effectiveness of coverage. It will usually be sufficient to model coverage as a single parameter for each fault type of interest rather than model the components of coverage.

As the system moves into engineering development, more model capabilities are required to better represent the operation of the hardware. More detailed investigation of recovery schemes and redundancy options may be used to assist engineers with design choices. Attention should also be given to the system reliability estimate. If it is inadequate, analytic models and methods can help identify those subsets of the system which cause the most failures. Fault trees and failure modes and effects analysis (FMEA) (references 18 and 19) can be used together to assist designers in identifying possible combinations of component failures which result in system failure and the effects of particular faults.

Analytic models and methods can support the certification process through more detailed analyses of the types noted above. Many input data values will be estimates. "Best case" and "worst case" analyses can be performed to determine bounds of system reliability for the ranges of the input values. The tools can be applied to individual subsystems of the flight control system. In addition, they can be applied to the entire system to evaluate interactions between subsystems and the use of the functional redundancy.

These analytical tools, such as CARSRA and fault trees, when used in conjunction with proper life cycle development phases, static, and dynamic software testing tools, error management and redundancy techniques, and configuration management practices, as outlined in RTCA DO-178 and other documents and texts, provide the basis of a methodology which, when properly applied, will result in the in-service implementation of digital flight control systems which should be: (a) Free from hazardous error, and (b) highly reliable.

CURRENT WORK.

The preceding material is discussed in greater detail in the "Handbook of Validation Processes for Digital Integrated Flight Control and Avionics Systems" developed for the FAA Technical Center by Battelle's Columbus Laboratories. The purpose of this handbook is to identify techniques, methodologies, tools, and procedures in a systems context that may be applicable to aspects of the validation and certification of digital systems at specific times in the development and certification portion of the system life cycle. The application of these techniques in the development of discrete units and/or systems will result in a completion of a product or system which is verifiable and can be validated in the context of the existing regulations/orders for the government regulatory agencies. The handbook uses a systems engineering approach to the integration and testing of software and hardware during the design, development, and implementation phases. The handbook also recognizes and provides for the evaluation of the pilot's work load and utilization of the new control/display technologies, especially when crew recognition and intervention may be necessary to cope with/recover from the effects of faults or failures in the digital systems, or the crew introduces errors into this system and the periods of high work load due to some inadvertent procedure or entry of incorrect or erroneous data.

In summary, the handbook:

(1) Identifies and presents the issues related to the design, development, and implementation of software based digital systems.

(2) Identifies specific approaches applicable to all aspects of the verification and validation procedures, at specific times in the development and certification portion of the system life cycle.

(3) Provides the government regulatory agencies (especially the Federal Aviation Administration (FAA) as well as the industry with a set of tools/procedures, in a systems engineering context which may be of value in the validation/certification process. This handbook is available from the FAA Technical Center.

In addition, the FAA Technical Center and Lockheed-Georgia are conducting an integrated assurance assessment of a digital flight control system. The purpose of this study is to "explore and demonstrate the integrated application of fault trees, reliability prediction and simulation in establishing that a representative digital flight control system is free of debilitating faults. The study will be conducted using an interactive logic tree graphics (fault tree) program, the automated reliability analysis program, CARSRA, and the Reconfigurable Digital Flight System Control (RDFCS) facility at NASA-Ames Research Center. The task summaries are presented in figure 7, table 9 and references 20 and 21.

TABLE 9. METHODS APPLICATION

	[illegible]	[illegible]	[illegible]
RELIABILITY	APPLICATION OF METHODS CARSRA TO THE DUAL-DUAL CONFIGURATION	HISTORICAL DATA ACQUISITION TO SUPPORT ANALYSIS	TRANSPARENCY AND PLAUSIBILITY OF MODELS
FAILURE EFFECTS	APPLICATION OF LOGIC TREE GRAPHICS TO THE DUAL-DUAL CONFIGURATION	REAL-TIME SYSTEM SIMULATION OF AUTOLAND FAILES	GENERALIZATIONS REGARDING FAULT CASES
PERFORMANCE	TEST CASE DEFINITION FOR DAS FAILS	PILOT-IN-THE-LOOP ASSESSMENT OF DEGRADATION	TEST PROCEDURE VALIDATION

The RDFCS is a representative fail operational/fail passive dual-dual configuration digital flight control system which closely resembles the Lockheed-Collins L-1011/500 fail operational Category IIIA autoland system. The simulator figures 8 and 9 is a system simulator designed for research utilization and for real-time closed-loop operation (reference 20).

At the same time, NASA-Ames, General Research Corporation, and Hughes Aircraft Company are conducting research in the development and quantitative analysis of automated static/dynamic verification tools applicable to digital flight control software. This research will also utilize the RDFCS facility. This research will concentrate on the assessment of error detection capabilities and cost effectiveness of the automated software tools, when applied to the DFCS software. The primary techniques to be used in this analysis will be error seeding in which the errors to be seeded will be representative in type and frequency of occurrence of those found during the development of the module, subsystem, and system software (reference 22).

A third effort underway between the FAA Technical Center, NASA-Ames and Lockheed-Georgia is the development and implementation of hardware fault insertion and instrumentation system (FIIS) which will allow circuit-card level, chip-level and pin level fault insertion capability under "robotic" or automated/controlled computer insertion timing and measurement of direct and latent effects. This facility, when completed will augment the existing manual circuit-card/chip-level fault insertion capability and will provide a more comprehensive estimate of coverage for the built-in test (BIT) and fault isolation test (FIT) software/hardware monitors, error detection schemes and redundancy management techniques utilized in the design development and implementation of the dual-dual digital flight control system.

CONCLUSIONS.

The development of an assurance assessment methodology for the verification and validation of a digital flight control system which is dependent on software of highest integrity and is highly reliable and capable of successful mission completion (including autoland, Category IIIA approach and landing) requires the integrated application of state-of-the-art tools and techniques. The application of these techniques and the understanding of the results by both industry and regulatory personnel should result in the in-service implementation of high technology DFCS which are safe and comply with the reliability requirements of existing advisory documents and regulatory documents for the current generation and perhaps the next generation of advanced digital flight control systems.

The results obtained from the ongoing laboratory work, the ongoing certification of current and next generation systems, and the technology advances in hardware/software integration may dictate that the existing tools and methodologies have to be augmented by new techniques such as finite state machines, Petri nets, and robotic testing in order to meet the high reliability and availability requirements.

REFERENCES

1. "Airplane System Design Analysis," Advisory Circular 25.1309-XX, Department of Transportation, Federal Aviation Administration, Federal Register, Vol. 46, Issue 214, November 5, 1981, Page 54958.

2. "The Safety Assessment of Systems," Subsection D-1, General and Definitions, British Civil Airworthiness Requirements, December 16, 1981, Civil Aviation Authority.

3. MIL-F-9490D, "Flight Control Systems Design, Installation, and Test of Piloted Aircraft, General Specifications," for U.S. Air Force, June 1975.

4. "Digital Fly-by-Wire Flight Control Validation Experience," Szalai, et al, NASA Technical Memorandum 72860, December 1978.

5. "Software Considerations in Airborne Systems and Equipment Certification," RTCA Document Number DO-178, November 1981.

6. "Comprehensive Methodologies for Digital Flight Control Software Verification," Mulcare and Kung, Proceeding of NAECON '80, Vol. 1, Pages 252-259, May 1980.

7. "The Cost of Developing Large-Scale Software," Wolverton, R. W., in Quantitative Management: Software Cost Estimating, IEEE Computer Society 1977, Pages 156-177.

8. "Sneak Circuit and Software Sneak Analysis," Goday, S. G., and Engels, G. J., Journal of Aircraft, Vol. 15, August 1978, Pages 509-513.

9. "Software Reliability," Myers, G. T., Wiley-Interscience Publication, 1976.

10. "Software Engineering," Jensen, R. W., and Tories, C. C., Prentice-Hall, Incorporated, 1979.

11. "Monitoring and Error Management in Computerized Systems," Hall, R., RTCA Paper No. 79-81/SC 145-26, February 20, 1981.

12. "Comparative Analysis of Techniques for Evaluating the Effectiveness of Aircraft Computing Systems," Hitt, et al, NASA Contractor Report CR 159358, April 1981.

13. " Fault-Tolerant System Reliability Modeling/ Analysis," Masreliez, J. C., and Bjurman, B. E., Journal of Aircraft, Vol. 14, No. 8, August 1977, Pages 803-808.

14. "Automated Reliability and Failure Effects Methods for Digital Flight Control and Avionic Systems: Volume I and II," Ness, et al, NASA Contractor Report CR 166148, March 1981.

15. "Industry Perspective on Simulation Methods and Research for Validation and Failure Effects Analysis of Advanced Digital Flight Control/ Avionics," Mulcare, et al, NASA Contractor Report CR 152234, February 1979.

16. "Digital Flight Control Software Validation Study," Rang, Gutmann, Mulcare, and Ness, AFFDL-TR-79-3076, June 1979.

17. "Simulator Investigation Plan for Digital Flight Controls Validation Technology," Mulcare, et al, Lockheed-Georgia Company Engineering Report No. LG81ER0126 (Prepared under NASA Contract NAS2-10270), May 1980.

18. "Fault Tree Handbook," Haasl, D. F. et al, NUREG-0492, U.S. Nuclear Regulatory Commission, January 1981.

19. "Fault/Failure Analysis Procedure," ARP-926B, Society of Automotive Engineers, Inc.

20. "Airborne Advanced Reconfigurable Computer System (ARCS)," Bjurman, B. E., et al, NASA CR-145024, August 1976.

21. "Software Reliability: Determination and Prediction," Gephart, et al, AFFDL-TR-78-77, June 1978.

22. "A Digital Flight Control System Verification Laboratory," De Feo, P., and Saib, S., NAECON, May 18-20, 1982.

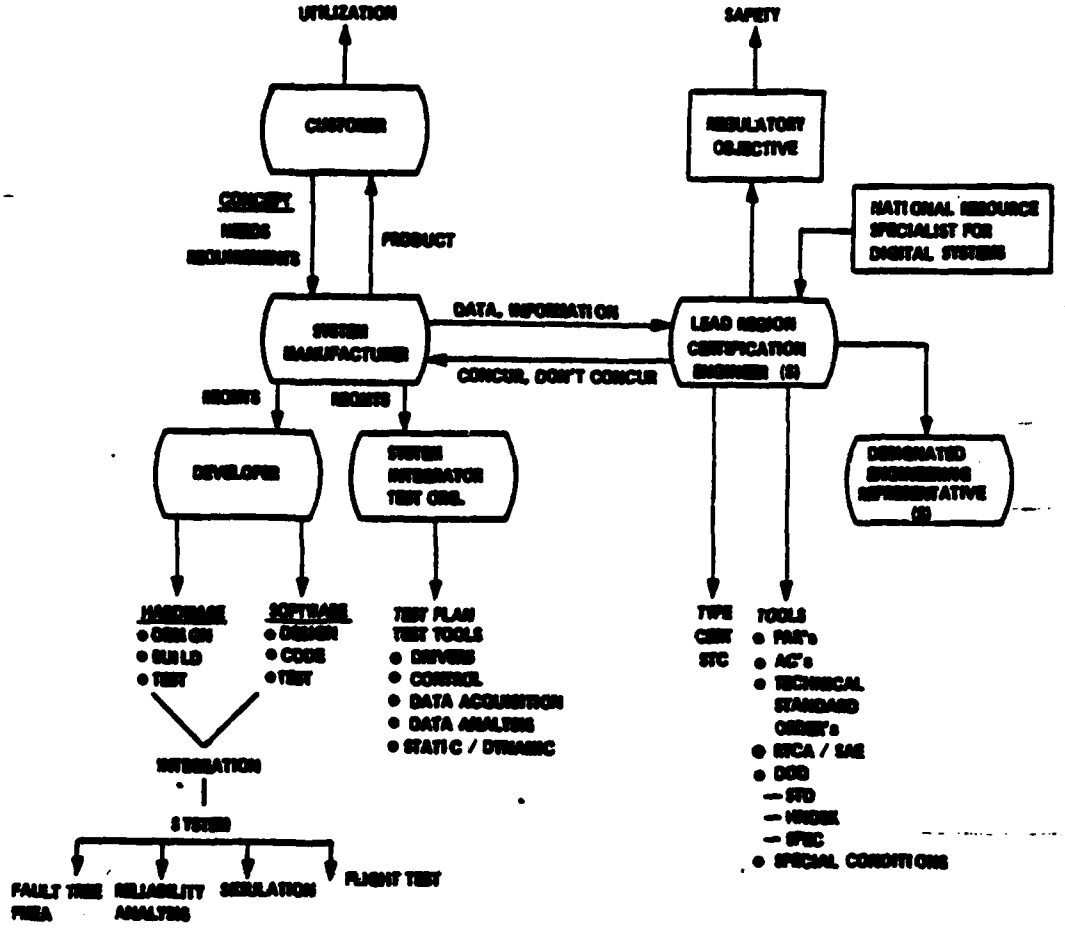


FIGURE 1. AGENTS/ROLES

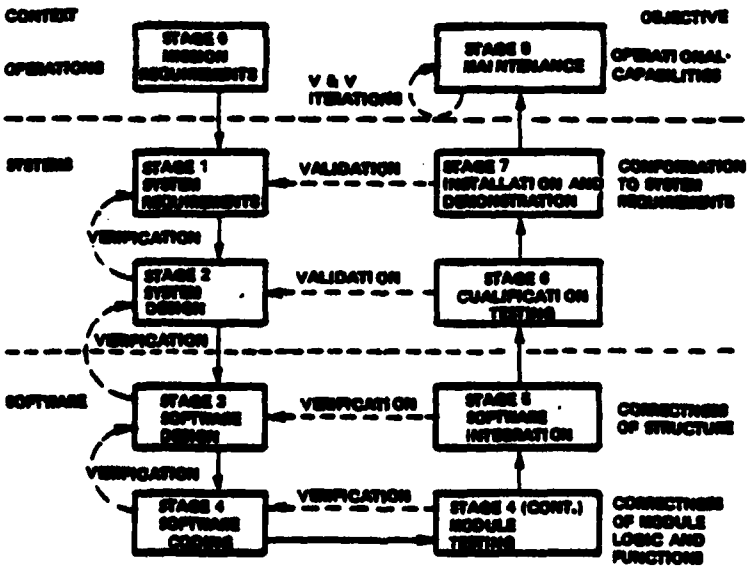


FIGURE 2. V&V THROUGHOUT THE SOFTWARE LIFE CYCLE

ACTIVITY \ DEVELOPMENT PHASE		PHASE A	PHASE B	PHASE C	PHASE D	PHASE E	PHASE F	PHASE G	PHASE H
		PERFORMANCE AND DESIGN REQUIREMENTS	IMPLEMENTATION CONCEPT AND TEST PLAN	INTERFACE & DATA REQUIREMENTS SPECIFICATION	DETAILED DESIGN SPECIFICATION	CODING AND AUDITING	SYSTEM TESTING	CERTIFICATION AND ACCEPTANCE	OPERATIONS AND MAINTENANCE
MANAGEMENT	1	PROGRAM MANAGEMENT	PROGRAM MANAGEMENT	PROGRAM MANAGEMENT	PROGRAM MANAGEMENT	PROGRAM MANAGEMENT	PROGRAM MANAGEMENT	PROGRAM MANAGEMENT	PROGRAM MANAGEMENT
	2	PROGRAM CONTROL	PROGRAM CONTROL	PROGRAM CONTROL	PROGRAM CONTROL	PROGRAM CONTROL	PROGRAM CONTROL	PROGRAM CONTROL	PROGRAM CONTROL
REVIEWS	3		PRELIMINARY DESIGN REVIEW (PDR)	INTERFACE DESIGN REVIEW (IDR)	CRITICAL DESIGN REVIEW (CDR)	SYSTEM TEST REVIEW (STR)		ACCEPTANCE TEST REVIEW (ATR)	OPERATIONAL CERTIFICATE
DOCUMENTS	4	DOCUMENT AND EDIT	DOCUMENT AND EDIT	DOCUMENT AND EDIT	DOCUMENT AND EDIT	DOCUMENT AND EDIT		DOCUMENT AND EDIT	DOCUMENT AND EDIT
	5	REPRODUCTION	REPRODUCTION	REPRODUCTION	REPRODUCTION	REPRODUCTION		REPRODUCTION	REPRODUCTION
REQUIREMENTS AND SPECIFICATIONS	6	REQUIREMENTS DEFINITION	FDR MATERIAL	EVENT GENERATION INTERFACE	PRODUCT CONFIG DETAILED TECH DESCRIPTION (PART I) (WITHOUT LISTINGS)	TECHNICAL DESCRIPTION UPDATE	PRODUCT CONFIG DETAILED TECH DESCRIPT UPDATE (PART II) (WITH LISTINGS)	REQUIREMENT CERTIFICATION	SOFTWARE PROBLEM REPORTS (SPR)
	7	REQUIREMENTS ALLOCATION	PERFORMANCE AND DESIGN REQUIREMENTS (PART I)	COMMAND DEFINITION I / F		TRAINING DOCUMENTATION		FINAL DOCUMENTATION UPDATE (PART II)	
	8			TELEMETRY DEFINITION I/F			INTERFACE SPECIFICATIONS UPDATE		
	9			OPERATIONAL ENVIRONMENT I/F					
DESIGN	10	TRADE STUDIES	TRADE STUDIES	TRADE STUDIES	TRADE STUDIES				
	11	INTERFACE REQUIREMENTS	FUNCTIONAL DEFINITION	DATA DEFINITION	ALGORITHM DESIGN	ALGORITHM UPDATE	PROGRAM UPDATE		
	12	HUMAN INTERACTION	STORAGE AND TIMING ALLOCATION		PROGRAM DESIGN				
	13	STANDARDS AND CONVENTIONS	DATA BASE DEFINITION	DATA BASE DESIGN		DATA BASE UPDATE	DATA BASE UPDATE		
	14		SOFTWARE OVERVIEW (PRELIMINARY)		SOFTWARE OVERVIEW UPDATE				
CODING	15				PROTOTYPE CODING	OPERATIONAL CODING			
TESTING, CONFIGURATION CONTROL, AND QAA	16		PRODUCT AND CONFIGURATION CONTROL	PRODUCT AND CONFIGURATION CONTROL	PRODUCT AND CONFIGURATION CONTROL	PRODUCT AND CONFIGURATION CONTROL	PRODUCT AND CONFIGURATION CONTROL	PRODUCT AND CONFIGURATION CONTROL	PRODUCT AND CONFIGURATION CONTROL
	17		DATA BASE CONTROL	DATA BASE CONTROL	DATA BASE CONTROL	DATA BASE CONTROL	DATA BASE CONTROL	DATA BASE CONTROL	DATA BASE CONTROL
	18	TEST REQUIREMENTS	TEST PLANS	INTERFACES	TEST PROCEDURES	DEVELOPMENT TESTING PLANNING	SYSTEM TEST PLANNING	ACCEPTANCE AND TEST PLANNING	INTEGRATION TESTING
	19					DEVELOPMENT TEST	SOFTWARE SYSTEM TEST	ACCEPTANCE DEMONSTRATION	SPR CLOSURE
	20						HARDWARE / SOFTWARE SYSTEM TESTING		
	21	ACCEPTANCE TEST REQUIREMENTS	QUALITY AND RELIABILITY ASSURANCE PLANS	QA AND SA MONITORING	QA AND SA MONITORING	Q AND SA MONITORING	Q AND SA MONITORING	Q AND SA MONITORING	Q AND SA MONITORING
	22					TEST SUPPORT	TEST SUPPORT	TEST SUPPORT	TEST SUPPORT
	23								
OPERATIONS	24		OPERATIONAL CONCEPT	OPERATIONAL TIMELINE	OPERATIONAL CONCEPT UPDATE	USER'S MANUAL (PRELIMINARY)	OPERATIONAL TIMELINE UPDATE	USER'S MANUAL UPDATE	INTEGRATION SUPPORT
	25		TRAINING PLAN		TRAINING PLAN UPDATE	TRAINING	TRAINING	TRAINING AND REHEARSAL	TRAINING AND REHEARSAL

FIGURE 3. SOFTWARE DEVELOPMENT PHASES AND ACTIVITIES

CLASS	SOURCES OF ERRORS OR FAILURES	TECHNIQUES FOR ERROR PREVENTION	TECHNIQUES FOR INFLIGHT ERROR AND FAILURE DETECTION AND / OR ISOLATION
1	• FAILURES IN HUMAN VISUAL ("MANUAL-SENSE") SYSTEMS OR SUBSYSTEMS		• FAILURE DETECTION USING VARIOUS FORMS OF BIT, AND / OR • MANUALLY REDUNDANT SUBSYSTEMS (WITH COMPARISON AND / OR VOTING)
2	• ERRORS IN COMPUTER / PROCESSOR HARDWARE OR MICROPROGRAM	• HARDWARE AND MICROPROGRAM • DESIGN DISCIPLINE • PROGRAMMING DISCIPLINE • VERIFICATION / VALIDATION • QUALITY CONTROL • CONFIGURATION MANAGEMENT	• MANUALLY REDUNDANT SUBSYSTEMS, PLUS: • DISSIMILAR HARDWARE • DISSIMILAR MICROPROGRAM
3	• ERRORS IN TRANSLATION FROM SOURCE CODE INTO OBJECT (MACHINE) CODE	• CLASS 2 TECHNIQUES, PLUS COMPILER / ASSEMBLER PROGRAM: • DESIGN DISCIPLINE • PROGRAMMING DISCIPLINE • VERIFICATION / VALIDATION • SAFETY • CONFIDENCE • QUALITY CONTROL • CONFIGURATION MANAGEMENT	• CLASS 2 TECHNIQUES, PLUS DISSIMILAR COMPILER / ASSEMBLER PROGRAMS
4	• ERRORS IN TRANSLATION FROM ALGORITHMS INTO SOURCE CODE	• CLASS 3 TECHNIQUES PLUS SOURCE CODE: • PROGRAMMING DISCIPLINE • VERIFICATION / VALIDATION • QUALITY CONTROL • CONFIGURATION MANAGEMENT	• CLASS 3 TECHNIQUES, PLUS DISSIMILAR SOURCE CODE
5	• ERRORS IN TRANSLATION FROM ENGINEERING DESIGN INTO ALGORITHMS	• CLASS 4 TECHNIQUES, PLUS ALGORITHMS: • DESIGN DISCIPLINE • VERIFICATION / VALIDATION	• CLASS 4 TECHNIQUES, PLUS DISSIMILAR OR VARIANT ALGORITHMS (CLASS 5 GOAL IS FAULT TOLERANT SUBSYSTEMS)
6	• ERRORS IN ENGINEERING DESIGN, i.e.: • CONCEPTS • APPLICATIONS • OBJECTIVES • SPECIFICATIONS • DERIVATIONS • STATEMENTS • DRAWINGS / FORMULAS • CODING, LABS • ASSUMPTIONS • APPROXIMATIONS • SIMPLIFICATIONS • LOGIC • MATHEMATICS • ETC.	• CLASS 5 TECHNIQUES, PLUS: • ENGINEERING DESIGN DISCIPLINE • COMPUTER SIMULATION / MODELING OF: • FUNCTIONAL (VISUAL, MANUAL, PANEL, ENGINEERING) • OTHER PERFORMANCE (INCLUDING ASSESSMENT OF STATISTICAL CORRELATE, IF APPROPRIATE) • GROUND AND FLIGHT TEST VALIDATION OF, AND CORRELATION WITH, THE FUNCTIONS	• DISPARATE MODES OR SENSORS "REVERSION" TO USE OF MANUAL ASSESSMENT SENSORS ASSISTED WITH SOME OR ALL CLASS 5 TECHNIQUES AS NECESSARY (CLASS 6 GOAL IS "ZERO FAIL-PASSIVE," OR "ZERO OR FAIL-OPERATIVE" SYSTEMS)

FIGURE 4. . MONITORING AND ERROR MANAGEMENT TECHNIQUES

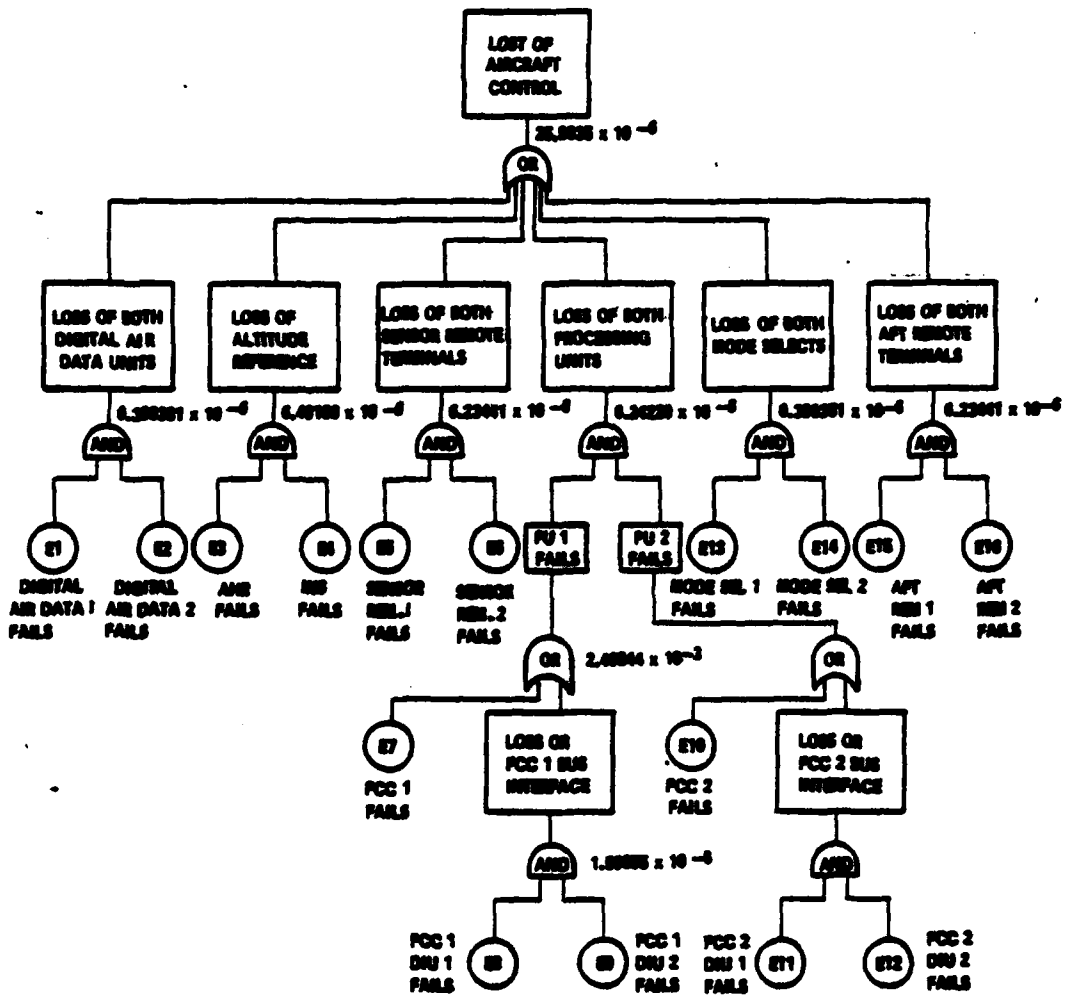
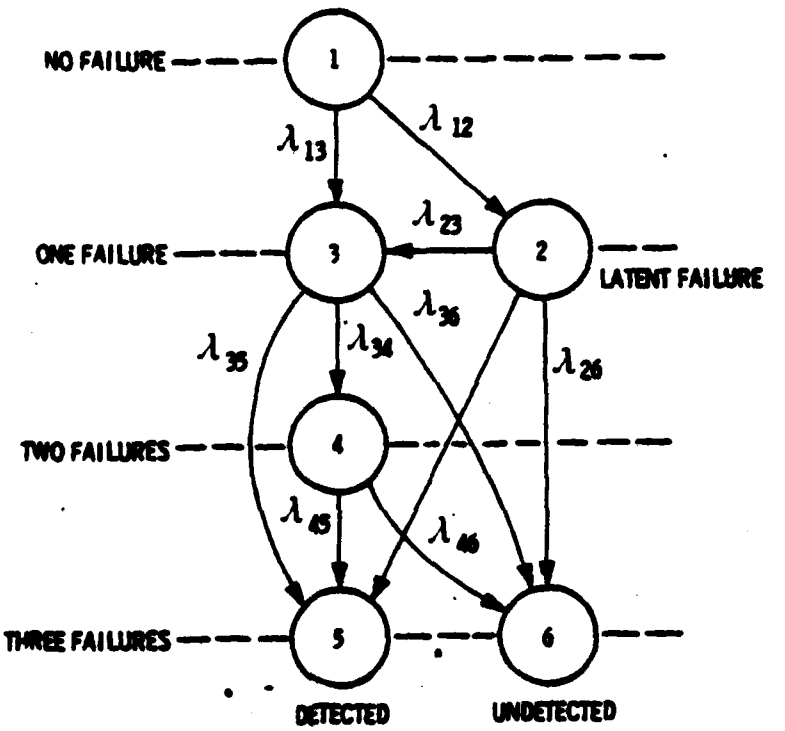


FIGURE 5. FAULT TREE FOR LOSS OF AIRCRAFT CONTROL



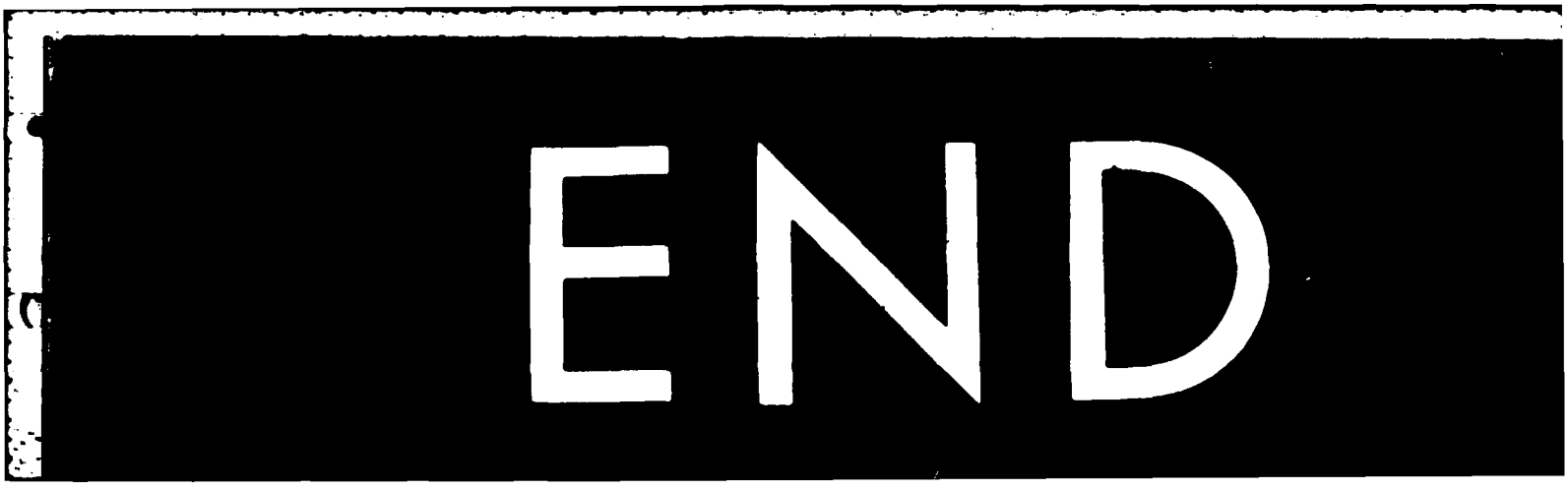
$$\text{EXAMPLE} - \lambda_{34} = 2\lambda_p \pi_{34} + 2\lambda_T \ell_{34}$$

π = PERMANENT RATE RATIO

ℓ = LEAKAGE

(REFERENCE 18)

FIGURE 6. EXAMPLE OF STAGE MARKOV MODEL


END